FOOD DELIVERY SYSTEM

A Mini Project Report submitted to

MOHAN BABU UNIVERSITY

in Partial Fulfillment of the Requirements for the Award of the degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

Submitted by

ARIKATLA SRIRAM	**(23102A030112)**
MUDE VAMSHI NAIK	**(23102A030114)**
GANTHI VENKATESH	**(23102A030139)**

Under the Guidance of

Mr. P. Yogendra Prasad
Assistant Professor
Department of Data Science

Department of Data Science

SCHOOL OF COMPUTING

MOHAN BABU UNIVERSITY

Sree Sainath Nagar, Tirupati – 517 102, A.P., INDIA

2024-25

MOHAN BABU UNIVERSITY

Sree Sainath Nagar, Tirupati – 517 102, A.P., INDIA
2024-25

DEPARTMENT OF DATA SCIENCE

CERTIFICATE

This is to certify that the mini project report entitled

"FOOD DELIVERY SYSTEM"

is the Bonafide work done by

ARIKATLA SRIRAM	**(23102A030112)**
MUDE VAMSHI NAIK	**(23102A030114)**
GANTHI VENKATESH	**(23102A030139)**

in the Department of **Data Science**, and submitted to Mohan Babu University, Tirupati in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering (Data Science) during the academic year 2024-2025. This work has been carried out under my supervision. The results of this mini project work havenot been submitted to any university for the award of any degree or diploma.

Guide:

Mr. P. Yogendra Prasad
Assistant Professor
Dept. of Data Science

Head:

Dr. P.K. Gupta
Professor & Head
Dept. of Data Science

INTERNAL EXAMINER

EXTERNALEXAMINER

DEPARTMENT OF DATA SCIENCE

Vision

To become a Centre of Excellence in Data Science by imparting high quality education through teaching, training and research

Mission

- To impart quality education in Computer Science and Engineering with specializations in Data Science by disseminating knowledge through contemporary curriculum, competent faculty and effective teaching-learning methodologies.
- Nurture research, innovation and entrepreneurial skills among students and faculty to contribute to the needs of industry and society.
- Inculcate professional attitude, ethical and social responsibilities for prospective and promising Engineering profession.
- Encourage students to engage in life-long learning by creating awareness of the contemporary developments in Computer Science and Engineering with specialization in Data Science.

PROGRAM EDUCATIONAL OBJECTIVES

After few years of graduation, the graduates of B. Tech. CSE(DS) will:

PEO1. Pursue higher studies in Computer Science, Data science or Management.

PEO2. Become successful entrepreneurs or be employed by acquiring required skill sets in the domains of Data Science and allied areas.

PEO3. Exhibit progression and effective adaptation to technological developments through life-long learning to address ever changing industrial requirements and follow ethical attitude in professional practice.

PROGRAM SPECIFIC OUTCOMES

On successful completion of the Program, the graduates of B. Tech. CSE(DS) program will be able to:

PSO1. Apply appropriate data analytical techniques for building effective decision-making systems.

PSO2. Develop intelligent systems using novel Machine Learning and Artificial Intelligence techniques.

PSO3. Design and develop efficient software systems using modern tools, techniques, and platforms to meet societal needs.

PSO4. Apply suitable tools and techniques to build secure distributed systems.

PROGRAM OUTCOMES

On successful completion of the Program, the graduates of B.Tech. CSE (DS) Program will be able to:

PO1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8. **Ethics**: Apply ethical principles and commit to professional ethics and

responsibilities and norms of the engineering practice.

PO9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

DECLARATION

We hereby declare that this project report titled **"FOOD DELIVERY SYSTEM"** is a genuine work carried out by us, in **B.Tech** ***(Computer Science and Engineering (Data Science)*** degree course of **Mohan Babu University, Tirupati** and has not been submitted to any other course or University for the award of any degree by us.

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea / data / fact / source in our submission. We understand that any violation of the above will cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Signature of the students

1.

2.

3.

ABSTRACT

In the modern era, technology has transformed various industries, with the food service sector being no exception. The increasing reliance on digital platforms has revolutionized food ordering and delivery, making it more accessible and convenient for customers. Traditionally, people had to visit restaurants or rely on phone orders, which was often inconvenient and time-consuming. However, with advancements in technology, online food delivery systems have emerged as a solution to streamline the ordering process.

A web-based food delivery system enables users to browse restaurant menus, place orders, and make payments seamlessly. To support this functionality, a robust database is required to store and manage critical information, including restaurant details, customer data, available food items, order history, delivery status, payment transactions, and customer feedback. This database ensures smooth operations by allowing data insertion, updates, and deletions as needed.

By integrating technology with the food service industry, this system enhances customer convenience, reduces operational inefficiencies, and provides real-time tracking of orders. As a result, online food delivery platforms have become an essential part of the modern lifestyle, catering to the growing demand for quick and efficient meal delivery services.
Keywords: SQL Server, Microsoft SQL server management studio, HTML, CSS, java script, PHP server.

TABLE OF CONTENTS

CHAPTER 1. INTRODUCTION

1.1 Introduction to the topic

In today's fast-paced world, technology has significantly transformed the way people order and receive food. Traditional food ordering methods, such as visiting restaurants or placing phone calls, were often time-consuming and inconvenient. To address these challenges, online food delivery systems have emerged as an efficient and user-friendly solution.

A food delivery system is a software-based platform that allows customers to browse menus, place orders, and make payments online. It automates essential processes such as order tracking, restaurant inventory management, and payment processing, reducing human errors and improving efficiency.

By providing a seamless experience, an online food delivery system enhances customer convenience, allowing them to explore multiple restaurant options and order meals without switching between different platforms. Additionally, automation in food delivery systems offers various business benefits, including real-time tracking, order management, and data analytics for better decision-making.

This system not only revolutionizes the food industry but also meets the growing demand for quick, efficient, and hassle-free meal delivery services.

1.2 Problem Statement

The **online food delivery system** is a dynamic web-based platform that enables users to browse restaurant menus, place orders, and make payments seamlessly. This system provides real-time details such as available restaurants, menu options, order status, delivery tracking, ratings, customer details, and payment history. It integrates various components, including restaurant information, customer and admin management, delivery services, payment processing, and receipt generation, ensuring a smooth and efficient ordering process.

To use the platform, users must first register by providing their personal details. Once successfully registered, they receive login credentials to access the system and place orders. Customers can select their preferred restaurant, browse the menu, and add food items to their cart. The system then checks the availability of the selected items and provides an estimated delivery time. If the selected items are available, the total cost, including taxes and delivery charges, is displayed. In case of unavailability, alternative suggestions are provided. Users can make secure payments through various options such as credit/debit cards, UPI, wallets, or net banking. After successful payment, an order receipt is generated, and customers can track their delivery in real time.

Designed with a user-friendly interface, the platform ensures easy navigation for users of all age groups. By automating the ordering process, managing real-time availability, and providing efficient delivery tracking, the system enhances customer convenience and business efficiency, making food ordering quick, hassle-free, and accessible from anywhere.

1.3 Objectives

The primary objective of the food delivery system is to create an efficient and user-friendly platform that enables customers to browse restaurant menus, place orders, and make payments seamlessly. The system ensures real-time availability tracking, allowing users to check food item availability, delivery slots, and restaurant services instantly. To enhance accessibility, the platform is designed with a simple and intuitive interface, making it easy to use for all age groups. Secure payment options, including credit/debit cards, UPI, wallets, and net banking, are integrated to provide flexibility and convenience during transactions.

Once an order is placed, the system generates instant order confirmation and enables real-time tracking, ensuring timely updates and transparency. Additionally, it maintains comprehensive records of restaurants, menus, admins, and delivery partners, streamlining operations and improving efficiency. The database also stores customer details, order history, and preferences, allowing for a personalized user experience. The system efficiently manages order bookings, modifications, and cancellations, ensuring hassle-free refunds and customer satisfaction.

To enhance service quality, the platform includes a ratings and feedback system, allowing customers to share their experiences and help improve restaurant performance. Furthermore, automation plays a crucial role in generating reports on sales, customer behavior, and restaurant efficiency, aiding in better business decision-making. Overall, this food delivery system aims to simplify online ordering, improve service efficiency, and provide a seamless experience for both customers and businesses.

CHAPTER 2. DATABASE DESIGN

2.1 List of Attributes, entities and relationships

1. **Entity Name: cuisines**

Attributes	Type
Cuisine_id	int(10)
name	varchar(50)

2. **Entity Name: users**

Attributes	Type
User_id	int(10)
Email	varchar(100)
password	varchar(100)
role	varchar(50)

3. **Entity Name: customers**

Attributes	Type
Customer_id	int(10)
User_id	Int(10)
name	Nvarchar(max)
phone	Varchar(10)
address	nvarchar(max)

4. **Entity Name: restaurants**

Attributes	Type
Restaurant_id	int(10)
User_id	int(10)
name	varchar(100)
description	Nvarchar(max)
address	Nvarchar(max)
contact	varchar(15)

5. **Entity Name: menu_items**

Attributes	Type
Item_id	int(10)
Restaurant_id	int(10)
name	Varchar(100)
description	nvarchar(max)

price	decimal
Image_url	Varchar(340)
Is_available	default

6. **Entity Name: payments**

Attributes	**Type**
Payment_id	int(10)
Order_id	Int(10)
amount	decimal
Payment_method	varchar(50)
Payment_status	varchar(50)
Payment_date	Datetime

7. **Entity Name: orders**

Attributes	**Type**
order_id	int(10)
Customer_id	int(10)
Restaurant_id	int(10)
Total_amount	Decimal
Status	Varchar(50)
Order_date	default
Payment_id	Int(10)

8. **Entity Name: order_items**

Attributes	**Type**
Order_items_id	int(10)
Order_id	int(10)
Item_id	Int(10)
Quantity	int(10)
Price	decimal

9. **Entity Name: delivery_agents**

Attributes	**Type**
Delivery_id	int(10)
User_id	int(10)
Name	Varchar(100)
Phone	Varchar(10)
Vehicle_details	Varchar(100)

10. **Entity Name: deliveries**

Attributes	**Type**
delivery_id	int(10)
Agent_id	int(10)

Order_id	Int(10)
Status	Varchar(50)
Estimated_delivery_time	datetime
Delivery_date	datetime

11. **Entity Name: reviews**

Attributes	**Type**
Review_id	int(10)
Customer_id	int(25)
Restaurant_id	int(10)
Delivery_id	int(10)
rating	int(10)
Comment	varchar(100)
Review_date	datetime

12. **Entity Name: wishlists**

Attributes	**Type**
Wishlist_id	int(10)
Customer_id	int25)
Item_id	int(10)

13. **Entity Name: coupons**

Attributes	**Type**
Coupon_id	int(10)
code	varchar(50)
Discount_percentage	decimal
Max_discount	decimal
Min_order_amount	Decimal
Expiry_date	date
Is_active	default

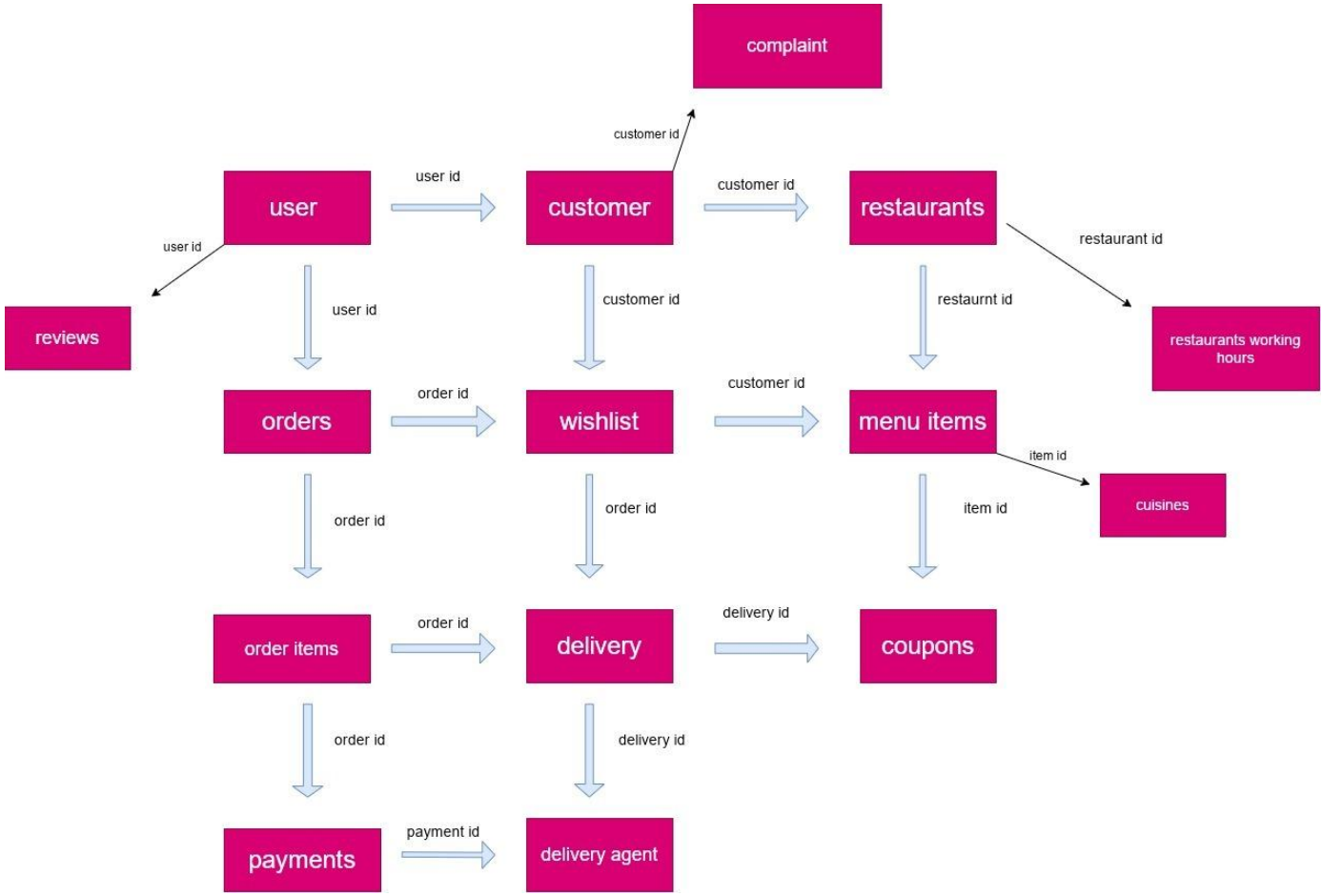
14. **Entity Name: complaints**

Attributes	**Type**
Complaint_id	int(10)
Order_id	int(10)
Customer_id	Int(10)
subject	varchar(100)
description	varchar(100)
status	varchar(100)
Complaint_date	default

15. **Entity Name: restaurant_working_hours**

Attributes	**Type**
Working_hour_id	int(10)
Restaurant_id	int(10)
Day_of_week	Varchar(100)
Opening_time	time
Closing_time	time
Is_closed	default

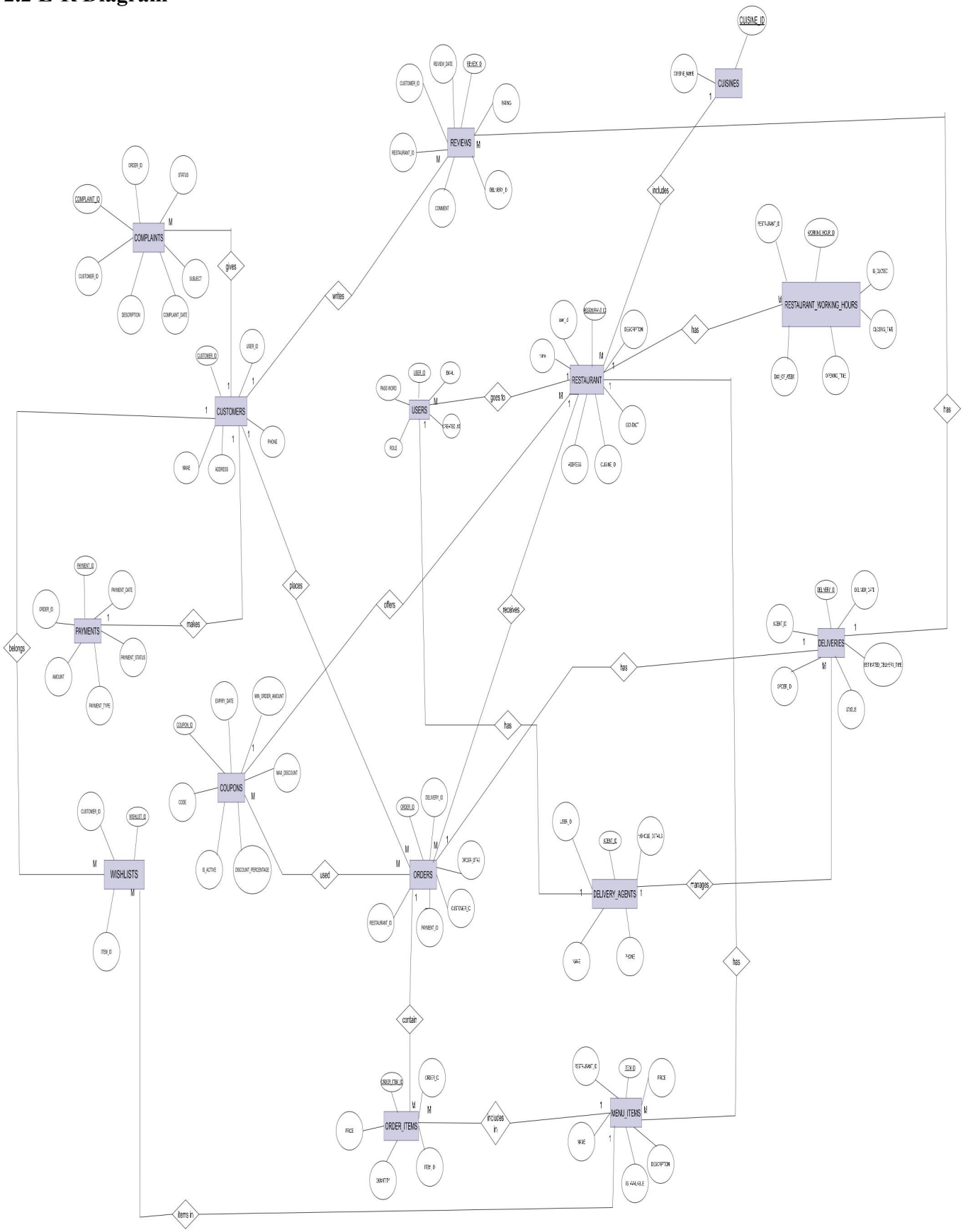
2.1.1 Entities and their relationships:



The above diagram is a simple representation of entities which shows the connectivity between all the entities and the relationship between various entities

To know in detail about the types of relationships that exist between all the entities and to know the different attributes that describes about the entity we design ER(entity relation) diagram.

2.2 E-R Diagram



CHAPTER 3. RELATIONAL MODEL

3.1 Database languages

Four categories of database languages :

1. Data definition language (DDL)

Data definition language (DDL) creates the framework of the database by specifying the database schema, which is the structure that represents the organization of data. Its common uses include the creation and alteration of tables, files, indexes and columns within the database. This language also allows users to rename or drop the existing database or its components.

Here's a list of DDL statements:

- CREATE: Creates a new database or object, such as a table, index or column.
- ALTER: Changes the structure of the database or object.
- DROP: Deletes the database or existing objects.
- RENAME: Renames the database or existing objects.

2. Data manipulation language (DML)

Data manipulation language (DML) provides operations that handle user requests, offering a way to access and manipulate the data that users store within a database. Its common functions include inserting, updating and retrieving data from the database.

Here's a list of DML statements:

- INSERT: Adds new data to the existing database table.
- UPDATE: Changes or updates values in the table.
- DELETE: Removes records or rows from the table.
- SELECT: Retrieves data from the table or multiple tables.

3. Data control language (DCL)

Data control language (DCL) controls access to the data that users store within a database. Essentially, this language controls the rights and permissions of the database system. It allows users to grant or revoke privileges to the database.

Here's a list of DCL statements:

- GRANT: Gives a user access to the database.
- REVOKE: Removes a user's access to the database.

4. Transaction control language (TCL)

Transaction control language (TCL) manages the transactions within a database. Transactions group a set of related tasks into a single, executable task. All the tasks must succeed in order for the transaction to work. Here's a list of TCL statements:

- COMMIT: Carries out a transaction.
- ROLLBACK: Restores a transaction if any tasks fail to execute.

3.2 Table Description

Following are the tables along with constraints used in All in one travel booking database.

1. company: This table contains various travel booking company details like company id, company name, description, moto of the company, status of the company, etc.

Constraint: company id should be provided if the company is linked up with this web site.

2. Administrator: Administrator entity contains the details of the administrator like admin id, admin name, login id, password.

Constraint: Here administrator id will not be accessed by any other relation because all the details of the tables and their relations, tuples will be accessed by the admin.

3. Customer: This table contains all the details of the customer such as customer id, name, date of birth, gender, address, email, login, password and also the status of the customer which described about the registration status of the customer.

Constraint: The customer id will be as the primary key constraint for the customer info relation in the database and also the customer id will be considered as the foreign key constraint for the other tuples like booking, receipt, cancellation, rating, etc.

4. Bus: This entity contains the detailed description about the bus like bus id, bus name, bus type, types of seats available in the bus and also the number of those seats count, availability of the bus.

Constraint: considering Bus id as the main constraint of the bus relation this id can be accessed by the boarding points relation where it contains the details of the bus whether it visits the particular boarding point or not.

5. *Boarding points:* The boarding point table describes about the number of buses passing through the particular point, it contains all the details of the bus that arrives and leaves the stopping and also the details regarding the cost of the type of seat selected from one boarding point to the other boarding point.

Constraint: Here we can consider the primary key as the boarding point id and the bus id as the foreign key constraint.

6. *Train:* The train table contains the details about the types of trains available, no. of trains available, train id, seating capacity of train, division of seats, arrival and departure timings of the train , name of the train and the cost of the seat.

Constraint: Train id will be the primary key constraint of this relation and to know for which company the train belongs to the company id would be considered as the foreign key.

7. *Flight:* The flight entity belongs to the service of the company provided , which deals with the details of the flight like flight id, flight name, location id's that flight lands and takes of at, arrival time and departure time of the flight , additional details like the no. of seats and division of those seats and the seating cost .

Constraint: Flight id will be the primary key constraint of this relation and to know for which company the train belongs to the company id would be considered as the foreign key.

8. *Booking:* The booking entity of the database deals with the no. of bookings held on the particular day, type of travel chosen , passenger name, ticket id, seat number, gender, contact and booking status .

Constraints: Booking id plays the role of primary key constraint in this relation and the receipt id would be referred as the foreign key constraint.

9. *Receipt:* Receipt relation contains the details about the bill generation on the booking of travel service by the customer, it will have the details regarding the receipt such as, receipt id, date , tax, billing cost, type of payment chosen, card number, and the status of the receipt.

Constraints: The receipt id will be referred as the primary key, and customer id will

be the foreign key to know that on whose id the bill was generated.

10. *Cancellation:* Cancellation table contains the details of the cancelled services request like cancellation id, receipt id, cancellation date, refundable amount, and the approval status of the cancellation.
Constraints: Cancellation id as the primary key would help in accessing the cancellation details of the receipts generated and to know this the receipt id will be considered as the foreign key.

11. *Rating*: Rating table will have the ratings of the customers given for the services the have been provided, to describe these details the attributes that are being constituted by the rating entity are rating id, customer id, company id, rating, comments title, comments, date of the rating and status of the rating.
Constraints: Rating id will be considered as the primary key constraint and the customer id will be used as the reference and will be a foreign key constraint of the relation.

12. *Location:* The location table contains the details like location id , location type, location title.
Constraints: Location id will be used as the reference to get all the details about the locations , so location id works as primary key in this relation.

13. *Services:* The service table will have the details about the types of services provided by the company . It have the attributes like service id, service type , cost of the service and availability of the service.
Constraint: Service id will be referenced as the primary key to give the details about the services available.

The above descripted information is the brief detailing about the entities and relations and their attributes.

3.3 Relational Database Scheme

The relational database schema for ***Food Delivery System*** database is as follows:

1. Cuisines(cuisineid,name)
2. Users(userid,email,password,role)
3. Customers(customerid,userid,name,phone,address)
4. Restaurants(restaurantid,userid,name,description,address,contact,cuisineid)
5. Menu_items(itemid,restaurantid,name,description,price,imageurl,isavailable)
6. Payments(paymentid,orderid,amount,paymentmethod,paymentstatus)
7. Orders(orderid,customerid,restaurantid,totalamount,status,orderdate,paymentid)
8. Order_items(orderitemid,orderid,itemid,quantityid,price)
9. Deliveryagents(agentid,userid,name,phone,vehicledetails)
10. Deliviries(deliveryid,agentid,orderid,status,estimateddeliverytime,deliverydate)
11. Reviews(reviewid,customerid,restaurantid,deliveryid,rating,comment,reviewdate)
12. Wishlists(wishlistid,customerid,itemid)
13. Coupons(couponid,code,discountpercent,maxdiscount,minorder amount,expiry date,isactive)
14. Complaints(complaintid,orderid,customerid,subject,description,status,complaint date)
15. Restaurant working hours(working hourid,restaurantid,day ofweek,openingtime,closingtime,isactive)

3.4 Relational Queries

```
CREATE DATABASE foodcourt;
-- Cuisines Table
CREATE TABLE Cuisines (
   cuisine_id INT IDENTITY(1,1) PRIMARY KEY,
   name VARCHAR(50) UNIQUE NOT NULL
);
INSERT INTO Cuisines (name) VALUES
('Italian'), ('Chinese'), ('Indian'), ('Mexican'), ('Thai'),
('French'), ('Japanese'), ('Korean'), ('Mediterranean'), ('Greek'),
('American'), ('Spanish'), ('Turkish'), ('Vietnamese'), ('Lebanese'),
('Brazilian'), ('British'), ('Caribbean'), ('African'), ('German');
select * from cuisines
-- Users Table
drop table Users;
CREATE TABLE Users (
   user_id INT IDENTITY(1,1) PRIMARY KEY,
   email VARCHAR(100) UNIQUE NOT NULL,
   password VARCHAR(100) NOT NULL,
   role VARCHAR(50) CHECK (role IN ('customer', 'restaurant', 'delivery', 'admin')) NOT NULL,
   created_at DATETIME DEFAULT GETDATE()
);
-- Since user_id is an IDENTITY column, you don't explicitly provide it during INSERT.
-- SQL Server will automatically generate the user_id values.
```

```
INSERT INTO Users (email, password, role) VALUES
('admin1@gmail.com', 'admin123', 'admin'),
('customer1@gmail.com', 'cust123', 'customer'),
('restaurant1@gmail.com', 'rest123', 'restaurant'),
('delivery1@gmail.com', 'del123', 'delivery'),
('customer2@gmail.com', 'cust456', 'customer'),
('admin2@gmail.com', 'admin456', 'admin');
-- Customers Table
CREATE TABLE Customers (
    customer_id INT IDENTITY(1,1) PRIMARY KEY,
    user_id INT UNIQUE,
    name VARCHAR(100) NOT NULL,
    phone VARCHAR(15),
    address NVARCHAR(MAX),
    FOREIGN KEY (user_id) REFERENCES Users(user_id) ON DELETE CASCADE
);
INSERT INTO Customers (user_id, name, phone, address) VALUES
(1, 'John Doe', '9876543210', 'Hyderabad, Telangana'),
(2, 'Priya Sharma', '8765432109', 'Bangalore, Karnataka'),
(3, 'Ravi Teja', '7654321098', 'Chennai, Tamil Nadu'),
(4, 'Sneha Kapoor', '6543210987', 'Mumbai, Maharashtra'),
(5, 'Vikram Singh', '5432109876', 'Pune, Maharashtra');
-- Restaurants Table
CREATE TABLE Restaurants (
    restaurant_id INT IDENTITY(1,1) PRIMARY KEY,
    user_id INT UNIQUE,
    name VARCHAR(100) NOT NULL,
    description NVARCHAR(MAX),
    address NVARCHAR(MAX),
    contact VARCHAR(15),
    cuisine_id INT,
    FOREIGN KEY (user_id) REFERENCES Users(user_id) ON DELETE CASCADE,
    FOREIGN KEY (cuisine_id) REFERENCES Cuisines(cuisine_id) ON DELETE SET NULL
);
INSERT INTO Restaurants (user_id, name, description, address, contact, cuisine_id) VALUES
(1, 'Tasty Bites', 'Serving the best fast food in town.', 'Hyderabad, Telangana', '9876543210', 1),
(2, 'The Spice Hub', 'Authentic Indian cuisine with a modern touch.', 'Mumbai, Maharashtra',
'8765432109', 3),
(3, 'Royal Biryani House', 'Delicious and mouth-watering biryani.', 'Chennai, Tamil Nadu', '7654321098',
```

```
3),
(4, 'Burger Express', 'Best burgers and fries in the city.', 'Delhi, New Delhi', '6543210987', 11),
(5, 'Pasta Palace', 'Italian food at its best.', 'Bangalore, Karnataka', '5432109876', 1);
-- Menu_Items Table
CREATE TABLE Menu_Items (
   item_id INT IDENTITY(1,1) PRIMARY KEY,
   restaurant_id INT,
   name VARCHAR(100) NOT NULL,
   description NVARCHAR(MAX),
   price DECIMAL(10,2) NOT NULL,
   image_url VARCHAR(255),
   is_available BIT DEFAULT 1,
   FOREIGN KEY (restaurant_id) REFERENCES Restaurants(restaurant_id) ON DELETE CASCADE
);
INSERT INTO Menu_Items (restaurant_id, name, description, price, image_url, is_available) VALUES
(1, 'Margherita Pizza', 'Classic Margherita with mozzarella cheese.', 299.99,
'https://example.com/pizza1.jpg', 1),
(1, 'Pepperoni Pizza', 'Loaded with pepperoni and cheese.', 349.99, 'https://example.com/pizza2.jpg', 1),
(1, 'Garlic Bread', 'Crispy garlic bread with butter.', 129.99, 'https://example.com/garlic.jpg', 1),
(2, 'Butter Chicken', 'North Indian Butter Chicken with naan.', 399.99,
'https://example.com/butterchicken.jpg', 1),
(2, 'Paneer Tikka', 'Paneer tikka with spicy masala.', 249.99, 'https://example.com/paneertikka.jpg', 1),
(3, 'Chicken Biryani', 'Hyderabadi Chicken Biryani with raita.', 499.99,
'https://example.com/chickenbiryani.jpg', 1),
(3, 'Mutton Biryani', 'Authentic Mutton Biryani.', 599.99, 'https://example.com/muttonbiryani.jpg', 1),
(3, 'Veg Biryani', 'Vegetarian Biryani with fresh vegetables.', 399.99,
'https://example.com/vegbiryani.jpg', 1),
(4, 'Chicken Burger', 'Grilled chicken patty with veggies.', 229.99, 'https://example.com/burger.jpg', 1),
(4, 'French Fries', 'Crispy french fries with ketchup.', 99.99, 'https://example.com/fries.jpg', 1),
(5, 'Pasta Alfredo', 'Creamy Alfredo pasta with cheese.', 319.99, 'https://example.com/pasta.jpg', 1),
(5, 'Lasagna', 'Layered lasagna with chicken and cheese.', 499.99, 'https://example.com/lasagna.jpg', 1),
(6, 'Chicken Fried Rice', 'Chinese style fried rice with chicken.', 269.99,
'https://example.com/friedrice.jpg', 1),
(6, 'Veg Hakka Noodles', 'Spicy Hakka noodles with veggies.', 199.99,
'https://example.com/noodles.jpg', 1),
(7, 'Amritsari Kulcha', 'Stuffed kulcha with chole masala.', 179.99, 'https://example.com/kulcha.jpg', 1),
(7, 'Paneer Butter Masala', 'Creamy paneer gravy with butter.', 329.99, 'https://example.com/paneer.jpg',
1),
(8, 'Tandoori Chicken', 'Tandoori grilled chicken with spices.', 499.99,
```

'https://example.com/tandoori.jpg', 1),
(8, 'Seekh Kebab', 'Grilled mutton seekh kebab.', 549.99, 'https://example.com/kebab.jpg', 1),
(9, 'Cheese Pizza', 'Classic cheese pizza with mozzarella.', 279.99, 'https://example.com/cheesepizza.jpg', 1),
(9, 'Chicken Wings', 'Spicy chicken wings with sauce.', 349.99, 'https://example.com/wings.jpg', 1),
(10, 'Masala Dosa', 'South Indian crispy dosa.', 159.99, 'https://example.com/dosa.jpg', 1),
(10, 'Idli Sambar', 'Soft idlis with sambar.', 129.99, 'https://example.com/idli.jpg', 1),
(11, 'Zinger Burger', 'Crispy chicken burger with mayo.', 299.99, 'https://example.com/zinger.jpg', 1),
(11, 'Popcorn Chicken', 'Fried chicken popcorn.', 179.99, 'https://example.com/popcorn.jpg', 1),
(12, 'Tacos', 'Mexican tacos with chicken filling.', 249.99, 'https://example.com/tacos.jpg', 1),
(12, 'Burrito', 'Stuffed burrito with beans and meat.', 399.99, 'https://example.com/burrito.jpg', 1),
(13, 'Sushi Roll', 'Japanese sushi roll with fish.', 499.99, 'https://example.com/sushi.jpg', 1),
(13, 'Ramen', 'Japanese noodle soup with chicken.', 399.99, 'https://example.com/ramen.jpg', 1),
(14, 'Hyderabadi Biryani', 'Authentic Hyderabadi biryani.', 549.99, 'https://example.com/hyderabadi.jpg', 1),
(14, 'Mirchi Bajji', 'Spicy chili fritters.', 99.99, 'https://example.com/bajji.jpg', 1),
(15, 'Fish Fry', 'Crispy fish fry with spices.', 399.99, 'https://example.com/fishfry.jpg', 1),
(15, 'Prawns Curry', 'Coastal prawns curry.', 499.99, 'https://example.com/prawns.jpg', 1),
(16, 'Chicken Curry', 'Spicy chicken curry.', 399.99, 'https://example.com/chickencurry.jpg', 1),
(16, 'Mutton Rogan Josh', 'Kashmiri Mutton Rogan Josh.', 599.99, 'https://example.com/roganjosh.jpg', 1),
(17, 'Chicken Domino Pizza', 'Chicken loaded pizza.', 499.99, 'https://example.com/pizza3.jpg', 1),
(17, 'Garlic Sticks', 'Garlic sticks with cheese.', 199.99, 'https://example.com/garlicsticks.jpg', 1),
(18, 'Chole Bhature', 'North Indian chole bhature.', 179.99, 'https://example.com/chole.jpg', 1),
(18, 'Samosa', 'Crispy samosa with chutney.', 59.99, 'https://example.com/samosa.jpg', 1),
(19, 'Submarine Sandwich', 'Subway-style sandwich.', 249.99, 'https://example.com/subway.jpg', 1),
(19, 'Paneer Wrap', 'Veg wrap with paneer.', 199.99, 'https://example.com/wrap.jpg', 1),
(20, 'Ghee Dosa', 'South Indian crispy dosa.', 199.99, 'https://example.com/gheedosa.jpg', 1),
(20, 'Uttapam', 'Thick South Indian pancake.', 149.99, 'https://example.com/uttapam.jpg', 1),
(21, 'Chicken Momos', 'Chinese style momos.', 149.99, 'https://example.com/momos.jpg', 1),
(21, 'Spring Roll', 'Crispy spring roll.', 99.99, 'https://example.com/springroll.jpg', 1),
(1, 'Chicken Supreme Pizza', 'Loaded with chicken chunks and cheese.', 399.99, 'https://example.com/supreme.jpg', 1),
(1, 'Veg Delight Pizza', 'Delightful veg pizza with capsicum and olives.', 299.99, 'https://example.com/vegpizza.jpg', 1),
(2, 'Malai Kofta', 'North Indian dish made from paneer and potatoes.', 349.99, 'https://example.com/malaikofta.jpg', 1),
(2, 'Dal Makhani', 'Creamy dal makhani with naan.', 279.99, 'https://example.com/dalmakhani.jpg', 1),
(3, 'Shawarma', 'Arabic Shawarma with chicken and mayo.', 199.99, 'https://example.com/shawarma.jpg',

```
1),
(3, 'Grilled Chicken', 'Grilled chicken with smoky spices.', 349.99, 'https://example.com/grilledchicken.jpg', 1),
(4, 'Chicken Manchurian', 'Chinese style chicken gravy.', 299.99, 'https://example.com/manchurian.jpg', 1),
(4, 'Paneer Chilli', 'Paneer cubes with spicy chilli sauce.', 249.99, 'https://example.com/paneerchilli.jpg', 1),
(5, 'Mutton Korma', 'Rich mutton curry with cashew paste.', 499.99, 'https://example.com/muttonkorma.jpg', 1),
(5, 'Chicken Korma', 'Spicy chicken korma with gravy.', 399.99, 'https://example.com/chickenkorma.jpg', 1),
(6, 'Dum Biryani', 'Hyderabadi Dum Biryani with spices.', 499.99, 'https://example.com/dumbiryani.jpg', 1);
-- Payments Table
CREATE TABLE Payments (
   payment_id INT IDENTITY(1,1) PRIMARY KEY,
   order_id INT UNIQUE,
   amount DECIMAL(10,2) NOT NULL,
   payment_method VARCHAR(50) CHECK (payment_method IN ('card', 'cash', 'wallet')) NOT NULL,
   payment_status VARCHAR(50) CHECK (payment_status IN ('pending', 'completed', 'failed')) NOT NULL,
   payment_date DATETIME DEFAULT GETDATE(),

);
INSERT INTO Payments (order_id, amount, payment_method, payment_status, payment_date) VALUES
(1, 399.99, 'card', 'completed', '2025-03-05 12:45:10'),
(2, 299.99, 'cash', 'completed', '2025-03-06 14:12:55'),
(3, 199.99, 'wallet', 'failed', '2025-03-06 18:22:30'),
(4, 499.99, 'card', 'completed', '2025-03-07 09:15:11'),
(5, 279.99, 'cash', 'completed', '2025-03-07 11:32:20');
-- Orders Table
CREATE TABLE Orders (
   order_id INT IDENTITY(1,1) PRIMARY KEY,
   customer_id INT NOT NULL,
   restaurant_id INT NOT NULL,
   total_amount DECIMAL(10,2) NOT NULL,
   status VARCHAR(50) NOT NULL CHECK (status IN ('pending', 'confirmed', 'preparing', 'out_for_delivery', 'delivered', 'cancelled')),
```

```
    order_date DATETIME DEFAULT GETDATE(),
    payment_id INT,
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id) ON DELETE CASCADE,
    FOREIGN KEY (restaurant_id) REFERENCES Restaurants(restaurant_id) ON DELETE NO ACTION, -- Corrected line
    FOREIGN KEY (payment_id) REFERENCES Payments(payment_id) ON DELETE SET NULL
);
INSERT INTO Orders (customer_id, restaurant_id, total_amount, status, order_date, payment_id) VALUES
(1, 5, 399.99, 'delivered', '2025-03-10 12:10:10', 1),
(2, 10, 189.99, 'cancelled', '2025-03-10 13:15:30', 2),
(3, 3, 299.99, 'delivered', '2025-03-10 14:20:45', 3),
(4, 15, 459.99, 'preparing', '2025-03-10 15:10:10', 4),
(5, 7, 159.99, 'out_for_delivery', '2025-03-10 16:20:30', 5);
-- Order_Items Table
CREATE TABLE Order_Items (
    order_item_id INT IDENTITY(1,1) PRIMARY KEY,
    order_id INT,
    item_id INT,
    quantity INT NOT NULL,
    price DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (order_id) REFERENCES Orders(order_id),
    FOREIGN KEY (item_id) REFERENCES Menu_Items(item_id)
);
INSERT INTO Order_Items (order_id, item_id, quantity, price) VALUES
(1, 5, 2, 299.99),
(2, 15, 1, 499.99),
(3, 22, 3, 199.99),
(4, 38, 2, 279.99),
(5, 47, 4, 399.99);
-- Delivery_Agents Table
CREATE TABLE Delivery_Agents (
    agent_id INT IDENTITY(1,1) PRIMARY KEY,
    user_id INT UNIQUE,
    name VARCHAR(100) NOT NULL,
    phone VARCHAR(15) not null,
    vehicle_details VARCHAR(100),
    FOREIGN KEY (user_id) REFERENCES Users(user_id) ON DELETE CASCADE
);
```

```
INSERT INTO Delivery_Agents (user_id, name, phone, vehicle_details) VALUES
(1, 'Rahul Sharma', '9876543210', 'Hero Splendor - KA05-6789'),
(2, 'Ajay Singh', '9876543221', 'Honda Activa - KA03-1234'),
(3, 'Manoj Kumar', '9876543232', 'Maruti Swift - AP29-8765'),
(4, 'Pavan Reddy', '9876543243', 'TVS Jupiter - TS10-3456'),
(5, 'Prakash Rao', '9876543254', 'Honda Dio - AP10-6543');
-- Deliveries Table
CREATE TABLE Deliveries (
   delivery_id INT IDENTITY(1,1) PRIMARY KEY,
   agent_id INT,
   order_id INT UNIQUE,
   status VARCHAR(50) CHECK (status IN ('assigned', 'picked_up', 'in_transit', 'delivered')) NOT
NULL,
   estimated_delivery_time DATETIME,
   delivery_date DATETIME,
   FOREIGN KEY (agent_id) REFERENCES Delivery_Agents(agent_id) ON DELETE SET NULL,
   FOREIGN KEY (order_id) REFERENCES Orders(order_id) ON DELETE NO ACTION
);
INSERT INTO Deliveries (agent_id, order_id, status, estimated_delivery_time, delivery_date)
VALUES
(5, 3, 'delivered', '2025-03-05 13:30:00', '2025-03-05 14:00:00'),
(12, 7, 'delivered', '2025-03-06 15:00:00', '2025-03-06 15:30:00'),
(21, 12, 'delivered', '2025-03-07 11:00:00', '2025-03-07 11:30:00'),
(32, 15, 'delivered', '2025-03-08 17:00:00', '2025-03-08 17:40:00'),
(11, 18, 'delivered', '2025-03-09 10:30:00', '2025-03-09 11:00:00');
-- Reviews Table
CREATE TABLE Reviews (
   review_id INT IDENTITY(1,1) PRIMARY KEY,
   customer_id INT,
   restaurant_id INT,
   delivery_id INT,
   rating INT CHECK (rating BETWEEN 1 AND 5),
   comment NVARCHAR(MAX),
   review_date DATETIME DEFAULT GETDATE(),
   FOREIGN KEY (customer_id) REFERENCES Customers(customer_id) ON DELETE CASCADE,
   FOREIGN KEY (restaurant_id) REFERENCES Restaurants(restaurant_id) ON DELETE NO
ACTION,
   );
drop table Reviews;
```

```
INSERT INTO Reviews (customer_id, restaurant_id, delivery_id, rating, comment, review_date) VALUES
(5, 3, 1, 5, 'Excellent service and fresh food!', '2025-03-05 14:10:00'),
(12, 7, 2, 4, 'Food was tasty, but delivery was late.', '2025-03-06 15:40:00'),
(21, 12, 3, 5, 'Loved the food! Will order again.', '2025-03-07 11:45:00'),
(32, 15, 4, 3, 'Food was cold upon delivery.', '2025-03-08 18:00:00'),
(11, 18, 5, 4, 'Quick delivery and good packaging.', '2025-03-09 11:20:00');
-- Wishlists Table
CREATE TABLE Wishlists (
   wishlist_id INT IDENTITY(1,1) PRIMARY KEY,
   customer_id INT,
   item_id INT,
   FOREIGN KEY (customer_id) REFERENCES Customers(customer_id) ON DELETE CASCADE,
   FOREIGN KEY (item_id) REFERENCES Menu_Items(item_id) ON DELETE NO ACTION
);
INSERT INTO Wishlists (customer_id, item_id) VALUES
(1, 5),
(2, 8),
(3, 10),
(4, 15),
(5, 20);
-- Coupons Table
CREATE TABLE Coupons (
   coupon_id INT IDENTITY(1,1) PRIMARY KEY,
   code VARCHAR(50) UNIQUE NOT NULL,
   discount_percent DECIMAL(5,2),
   max_discount DECIMAL(10,2),
   min_order_amount DECIMAL(10,2),
   expiry_date DATE,
   is_active BIT DEFAULT 1
);
INSERT INTO Coupons (code, discount_percent, max_discount, min_order_amount, expiry_date, is_active) VALUES
('SUMMER10', 10.00, 50.00, 200.00, '2025-06-30', 1),
('FESTIVE20', 20.00, 100.00, 300.00, '2025-12-31', 1),
('WELCOME5', 5.00, 20.00, 100.00, '2025-05-15', 1),
('NEWYEAR25', 25.00, 150.00, 500.00, '2025-01-01', 1),
('HOLIDAY15', 15.00, 60.00, 250.00, '2025-08-15', 1);
CREATE TABLE Complaints (
```

```
    complaint_id INT IDENTITY(1,1) PRIMARY KEY,
    order_id INT,
    customer_id INT NOT NULL,
    subject NVARCHAR(MAX),
    description NVARCHAR(MAX),
    status VARCHAR(50) CHECK (status IN ('open', 'resolved', 'closed')),
    complaint_date DATETIME DEFAULT GETDATE(),
    FOREIGN KEY (order_id) REFERENCES Orders(order_id) ON DELETE NO ACTION,
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id) ON DELETE CASCADE
);
INSERT INTO Complaints (order_id, customer_id, subject, description, status, complaint_date) VALUES
(1, 5, 'Late Delivery', 'The order was delivered 45 minutes late and the food was cold.', 'open', '2025-03-01 14:00:00'),
(2, 12, 'Cold Food', 'The food arrived cold and was not fresh at all.', 'resolved', '2025-03-02 16:00:00'),
(3, 21, 'Missing Item', 'One item from the order was missing. Please ensure all items are delivered next time.', 'open', '2025-03-03 12:00:00'),
(4, 32, 'Wrong Item', 'Received a wrong item in my order. I ordered a pizza but got pasta.', 'resolved', '2025-03-04 13:00:00'),
(5, 11, 'Food Quality', 'The food quality was subpar and tasted stale.', 'closed', '2025-03-05 10:00:00');
-- Restaurant_Working_Hours Table
CREATE TABLE Restaurant_Working_Hours (
    working_hour_id INT IDENTITY(1,1) PRIMARY KEY,
    restaurant_id INT,
    day_of_week VARCHAR(50) CHECK (day_of_week IN ('Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday')) NOT NULL,
    opening_time TIME,
    closing_time TIME,
    is_closed BIT DEFAULT 0,
    FOREIGN KEY (restaurant_id) REFERENCES Restaurants(restaurant_id) ON DELETE CASCADE
);
INSERT INTO Restaurant_Working_Hours (restaurant_id, day_of_week, opening_time, closing_time, is_closed) VALUES
(1, 'Monday', '08:00:00', '22:00:00', 0),
(1, 'Tuesday', '08:00:00', '22:00:00', 0),
(1, 'Wednesday', '08:00:00', '22:00:00', 0),
(1, 'Thursday', '08:00:00', '22:00:00', 0),
(1, 'Friday', '08:00:00', '23:00:00', 0),
(1, 'Saturday', '10:00:00', '23:00:00', 0),
```

```
(1, 'Sunday', '10:00:00', '22:00:00', 0),
(2, 'Monday', '09:00:00', '21:00:00', 0),
-- Query 1. Insert a new cuisine
INSERT INTO Cuisines (name) VALUES ('andhra');
-- Query 2. Retrieve all cuisines
SELECT * FROM Cuisines;
-- Query 3. Update a cuisine name
UPDATE Cuisines SET name = 'hyderabadhi' WHERE cuisine_id = 3;
select * from cuisines
-- Query 4. Delete a cuisine
DELETE FROM Cuisines WHERE cuisine_id = 2;
-- Query 5. Count total cuisines
SELECT COUNT(*) as total_items FROM Cuisines;
-- Query 6. Insert a new user
INSERT INTO Users (email, password, role) VALUES ('user@example.com', 'securepassword',
'customer');
-- Query 7. Retrieve all users
SELECT * FROM Users;
-- Query 8. Update user role
UPDATE Users SET role = 'admin' WHERE user_id = 5;
select*from users;
-- Query 9. update a user
select * from users
update users set password = 'venkyjagan123'
where email = 'admin1@gmail.com';
-- Query 10. Count total users
SELECT COUNT(*)as total FROM Users;
-- Query 11. Insert a new customer
INSERT INTO Customers (user_id, name, phone, address) VALUES (51, 'John Doe', '1234567890',
'ap');
-- Query 12. Retrieve all customers
SELECT * FROM Customers;
-- Query 13. Update customer address
UPDATE Customers SET address = 'kurnool,ap' WHERE customer_id = 1;
select*from customers;
-- Query 14. update a customer adress
update customers set address ='Hyderabad, Telangana'
where user_id=2;
select * from customers;
```

-- **Query** 15. Count total customers

SELECT COUNT(*) as total_customer FROM Customers;

-- **Query** 16. Insert a new restaurant

INSERT INTO Restaurants (user_id, name, description, address, contact, cuisine_id) VALUES (51, 'Pasta House', 'Italian Cuisine', 'hyd,telengana', '0987654321', 1);

-- **Query** 17. Retrieve all restaurants

SELECT * FROM Restaurants;

-- **Query** 18. Update restaurant description

UPDATE Restaurants SET description = 'it is very hyginic' WHERE restaurant_id = 9;

select*from restaurants;

-- **Query** 19.user id count from resturants

select count(user_id)as total from Restaurants where user_id = 1;

-- **Query** 20. Count total restaurants

SELECT COUNT(*)as total FROM Restaurants;

-- **Query** 21. Insert a new menu item

INSERT INTO Menu_Items (restaurant_id, name, description, price, image_url) VALUES (26, 'Spaghetti', 'Delicious spaghetti with marinara sauce', 12.99, 'http://example.com/spaghetti.jpg');

-- **Query** 22. Retrieve all menu items

SELECT * FROM Menu_Items;

-- **Query** 23. Update menu item price

UPDATE Menu_Items SET price = 13.99 WHERE item_id = 21;

select * from menu_items

-- **Query** 24. Delete a menu item

select avg(price) as total

from menu_items

where is_available = 1;

-- **Query** 25. Count total menu items

SELECT COUNT(*)as total FROM Menu_Items;

-- **Query** 26. Insert a new payment

INSERT INTO Payments (order_id, amount, payment_method, payment_status) VALUES (101, 25.99, 'card', 'completed');

-- **Query** 27. Retrieve all payments

SELECT * FROM Payments;

-- **Query** 28. Update payment status

UPDATE Payments SET payment_status = 'failed' WHERE payment_id = 1;

select*from payments

-- **Query** 29. Delete a payment

DELETE FROM Payments WHERE payment_id = 2;

-- **Query** 30. Count total payments

```
SELECT COUNT(*)as total FROM Payments;
-- Query 31. Insert a new order
INSERT INTO Orders (customer_id, restaurant_id, total_amount, status) VALUES (1, 1, 25.99, 'pending');
-- Query 32. Retrieve all orders
SELECT * FROM Orders;
-- Query 33. Update order status
UPDATE Orders SET status = 'delivered' WHERE order_id = 100;
select*from orders
-- Query 34. total amount from an order
select distinct  total_amount
from orders
--Query 35. Count total orders
SELECT COUNT(*)as total FROM Orders;
-- Query 36. Insert a new order item
INSERT INTO Order_Items (order_id, item_id, quantity, price) VALUES (1, 1, 27, 24.99);
select*from order_items
-- Query 37. Retrieve all order items
SELECT * FROM Order_Items;
-- Query 38. Update order item quantity
UPDATE Order_Items SET quantity = 3 WHERE order_item_id = 8;
select*from order_items
-- Query 39. Delete an order item
DELETE FROM Order_Items WHERE order_item_id = 2;
-- Query 40. Count total order items
SELECT COUNT(*)as total FROM Order_Items;
-- Query 41. Insert a new delivery agent
INSERT INTO Delivery_Agents (user_id, name, phone, vehicle_details) VALUES (51, 'Alice Smith', '1234567890', 'tvs-Bike');
-- Query 42. Retrieve all delivery agents
SELECT * FROM Delivery_Agents;
-- Query 43. Update delivery agent phone
UPDATE Delivery_Agents SET phone = '0787878781' WHERE agent_id = 41;
-- Query 44. Delete a delivery agent
DELETE FROM Delivery_Agents WHERE agent_id = 5;
-- Query 45. Count total delivery agents
SELECT COUNT(*) as total FROM Delivery_Agents;
-- Query 46. Insert a new delivery
INSERT INTO Deliveries (agent_id, order_id, status) VALUES (41, 1, 'assigned');
```

-- **Query** 47. Retrieve all deliveries

SELECT * FROM Deliveries;

-- **Query** 48. Update delivery status

UPDATE Deliveries SET status = 'delivered' WHERE delivery_id = 1;

select*from deliveries

-- **Query** 49. Delete a delivery

DELETE FROM Deliveries WHERE delivery_id = 8;

-- **Query** 50. Count total deliveries

SELECT COUNT(*)as total FROM Deliveries;

-- **Query** 51. Insert a new review

INSERT INTO Reviews (customer_id, restaurant_id, delivery_id, rating, comment) VALUES (1, 1, 1, 5, 'Excellent food!');

-- **Query** 52. Retrieve all reviews

SELECT * FROM Reviews;

-- **Query** 53. Update review rating

UPDATE Reviews SET rating = 4 WHERE review_id = 6;

select*from reviews

-- **Query** 54. Delete a review

DELETE FROM Reviews WHERE review_id = 7;

-- **Query** 55. Count total reviews

SELECT COUNT(*) as total FROM Reviews;

-- **Query** 56. Insert a new wishlist item

INSERT INTO Wishlists (customer_id, item_id) VALUES (1, 1);

-- **Query** 57. Retrieve all wishlist items

SELECT * FROM Wishlists;

-- **Query** 58. Delete a wishlist item

DELETE FROM Wishlists WHERE wishlist_id = 38;

-- **Query** 59. Count total wishlist items

SELECT COUNT(*)as total FROM Wishlists;

-- **Query** 60. Insert a new coupon

INSERT INTO Coupons (code, discount_percent, max_discount, min_order_amount, expiry_date) VALUES ('SAVE10', 10.00, 5.00, 20.00, '2023-12-31');

-- **Query** 61. Retrieve all coupons

SELECT * FROM Coupons;

-- **Query** 62. Update coupon status

UPDATE Coupons SET is_active = 0 WHERE coupon_id = 31;

select * from coupons

-- **Query** 63. Delete a coupon

DELETE FROM Coupons WHERE coupon_id = 10;

-- **Query** 64. Count total coupons

SELECT COUNT(*)as total FROM Coupons;

-- **Query** 65. Insert a new complaint

INSERT INTO Complaints (order_id, customer_id, subject, description) VALUES (1, 1, 'Order not received', 'I have not received my order yet.');

-- **Query** 66. Retrieve all complaints

SELECT * FROM Complaints;

-- **Query** 67. Update complaint status

UPDATE Complaints SET status = 'resolved' WHERE complaint_id = 30;

select * from complaints

-- **Query** 68. Delete a complaint

DELETE FROM Complaints WHERE complaint_id = 42;

-- **Query** 69. Count total complaints

SELECT COUNT(*)as total FROM Complaints;

-- **Query** 70. Insert new working hours for a restaurant

INSERT INTO Restaurant_Working_Hours (restaurant_id, day_of_week, opening_time, closing_time) VALUES (1, 'Monday', '09:00:00', '22:00:00');

-- **Query** 71. Retrieve all working hours

SELECT * FROM Restaurant_Working_Hours;

-- **Query** 72. Update working hours

UPDATE Restaurant_Working_Hours SET closing_time = '23:00:00' WHERE working_hour_id = 55;

select * from restaurant_working_hours

-- **Query** 73. Delete working hours

DELETE FROM Restaurant_Working_Hours WHERE working_hour_id = 13;

-- **Query** 74. Count total working hours entries

SELECT COUNT(*)as total FROM Restaurant_Working_Hours;

-- **Query** 75. Retrieve all orders with customer details

SELECT o.order_id, c.name, o.total_amount FROM Orders o JOIN Customers c ON o.customer_id = c.customer_id;

-- **Query** 76. Retrieve all menu items for a specific restaurant

SELECT m.name FROM Menu_Items m JOIN Restaurants r ON m.restaurant_id = r.restaurant_id WHERE r.name = 'Pasta House';

-- **Query** 77. Retrieve all reviews for a specific restaurant

select distinct rating from reviews

-- **Query** 78. Retrieve total sales for each restaurant

SELECT r.name, SUM(o.total_amount) AS total_sales FROM Restaurants r JOIN Orders o ON r.restaurant_id = o.restaurant_id GROUP BY r.name;

-- **Query** 79. Retrieve all customers who have placed an order

SELECT DISTINCT c.name FROM Customers c JOIN Orders o ON c.customer_id = o.customer_id;

-- **Query** 80. Count total orders by status

```
SELECT status, COUNT(*) FROM Orders GROUP BY status;
```

-- **Query** 81. Average rating for each restaurant

```
SELECT r.name, AVG(re.rating) AS average_rating FROM Restaurants r JOIN Reviews re ON r.restaurant_id = re.restaurant_id GROUP BY r.name;
```

-- **Query** 82. Total number of deliveries by agent

```
SELECT da.name, COUNT(d.delivery_id) AS total_deliveries FROM Delivery_Agents da LEFT JOIN Deliveries d ON da.agent_id = d.agent_id GROUP BY da.name;
```

-- **Query** 83. Total revenue generated from payments

```
SELECT SUM(amount) AS total_revenue FROM Payments;
```

-- **Query** 84. Count of active coupons

```
SELECT COUNT(*) as total FROM Coupons WHERE is_active = 1;
```

-- **Query** 85. Retrieve all customers with more than one order

```
SELECT c.name FROM Customers c JOIN Orders o ON c.customer_id = o.customer_id GROUP BY c.name HAVING COUNT(o.order_id) > 1;
```

-- **Query** 86. Retrieve all restaurants that are currently open

```
SELECT r.name FROM Restaurants r JOIN Restaurant_Working_Hours wh ON r.restaurant_id = wh.restaurant_id WHERE wh.is_closed = 0;
```

-- **Query** 87. Retrieve all menu items that are available

```
SELECT * FROM Menu_Items WHERE is_available = 1;
```

-- **Query** 88. Retrieve all complaints that are still open

```
SELECT * FROM Complaints WHERE status = 'open';
```

-- **Query** 89. Retrieve all orders placed in the last 30 days

```
SELECT * FROM Orders WHERE order_date >= DATEADD(DAY, -30, GETDATE());
```

-- **Query** 90. Retrieve all orders along with payment details

```
SELECT o.order_id, p.amount, p.payment_status FROM Orders o LEFT JOIN Payments p ON o.payment_id = p.payment_id;
```

-- **Query** 91. Retrieve all customers who have left reviews

```
SELECT DISTINCT c.name FROM Customers c JOIN Reviews r ON c.customer_id = r.customer_id;
```

-- **Query** 92. Retrieve all menu items with their restaurant names

```
SELECT m.name AS item_name, r.name AS restaurant_name FROM Menu_Items m JOIN Restaurants r ON m.restaurant_id = r.restaurant_id;
```

-- **Query** 93. Retrieve all deliveries for a specific order

```
SELECT d.delivery_id, d.status FROM Deliveries d WHERE d.order_id = 1;
```

-- **Query** 94. Retrieve the highest-rated restaurant

```
SELECT r.name FROM Restaurants r JOIN Reviews re ON r.restaurant_id = re.restaurant_id GROUP BY r.name ORDER BY AVG(re.rating) DESC;
```

-- **Query** 95. Retrieve all customers with their wishlist items

```
SELECT c.name, w.item_id FROM Customers c JOIN Wishlists w ON c.customer_id = w.customer_id;
```

```
-- Query 96. Retrieve all complaints related to a specific order
SELECT * FROM Complaints WHERE order_id = 2;
-- Query 97. Retrieve all active coupons that can be applied to an order
SELECT * FROM Coupons WHERE is_active = 1 AND min_order_amount <= 25.00;
-- Query 98. Retrieve all delivery agents who have completed deliveries
SELECT da.name FROM Delivery_Agents da JOIN Deliveries d ON da.agent_id = d.agent_id WHERE
d.status = 'delivered';
-- Query 99. Retrieve all restaurants that serve a specific cuisine
SELECT r.name FROM Restaurants r JOIN Cuisines c ON r.cuisine_id = c.cuisine_id WHERE c.name
= 'Italian';
-- Query 100. Retrieve all orders with their respective delivery agents
SELECT o.order_id, da.name AS delivery_agent FROM Orders o LEFT JOIN Deliveries d ON
o.order_id = d.order_id LEFT JOIN Delivery_Agents da ON d.agent_id = da.agent_id;
-- Query101. Retrieve Customers Who Have Ordered from a Specific Restaurant
SELECT name FROM customers
WHERE customer_id IN (
   SELECT DISTINCT customer_id FROM orders
   WHERE restaurant_id = (SELECT restaurant_id FROM restaurants WHERE name = 'Pizza Palace')
);
-- Query 102.Retrieve Orders Containing a Specific Menu Item
SELECT order_id FROM orders
WHERE order_id IN (
   SELECT order_id FROM order_items
   WHERE item_id = (SELECT item_id FROM menu_items WHERE name = 'Burger')
);
-- Query103.. Retrieve All Orders Paid with Credit Cards
SELECT * FROM orders
WHERE order_id IN (
   SELECT order_id FROM payments WHERE payment_method = 'credit_card' AND status = 'paid'
);
-- Query 104. Find Customers Who Have Not Placed Any Orders
SELECT name FROM customers
WHERE customer_id NOT IN (
   SELECT DISTINCT customer_id FROM orders
);
-- Query 105.. Find the Most Expensive Menu Item for Each Restaurant
SELECT mi.name ,r.name,price FROM menu_items mi
JOIN restaurants r ON mi.restaurant_id = r.restaurant_id
WHERE price = (
```

```
    SELECT MAX(price) FROM menu_items
    WHERE restaurant_id = mi.restaurant_id
);
-- Query 106. Retrieve the Highest-Rated Restaurants
SELECT name FROM restaurants
WHERE restaurant_id IN (
    SELECT restaurant_id FROM reviews
    WHERE rating = (SELECT MAX(rating) FROM reviews)
);
-- corelated sub queries
-- Query 107.Retrieve Customers Who Have Placed More Than 2 Orders
SELECT name FROM customers c
WHERE (SELECT COUNT(*) FROM orders o WHERE o.customer_id = c.customer_id) > 2;
-- Query 108. Retrieve Orders Where Total Amount Is Greater Than the Average Order Amount
SELECT order_id, total_amount FROM orders o
WHERE total_amount > (SELECT AVG(total_amount) FROM orders);
-- Query 109. Find Restaurants That Have Complaints Associated with Their Orders:
SELECT r.name
FROM Restaurants r
WHERE EXISTS (
    SELECT 1
    FROM Orders o
    JOIN Complaints c ON o.order_id = c.order_id
    WHERE o.restaurant_id = r.restaurant_id
);
-- Query 110.. Find Customers Who Have Items in Their Wishlist That Are Pricier Than 400:
SELECT c.name
FROM Customers c
WHERE EXISTS (
    SELECT 1
    FROM Wishlists w
    JOIN Menu_Items mi ON w.item_id = mi.item_id
    WHERE w.customer_id = c.customer_id
      AND mi.price > 400
);
-- Query 111. Find Orders Where the Total Amount Exceeds the Average Order Amount for the Same
Customer
SELECT o.order_id, o.total_amount
FROM Orders o
```

```
WHERE o.total_amount > (
   SELECT AVG(total_amount)
   FROM Orders
   WHERE customer_id = o.customer_id
);
-- Query 112.Find Delivery Agents Who Have Delivered Orders with a 'delivered' Status
SELECT da.name
FROM Delivery_Agents da
WHERE EXISTS (
   SELECT 1
   FROM Deliveries d
   WHERE d.agent_id = da.agent_id
     AND d.status = 'delivered'
);
-- Query 113.Find Restaurants That Have Menu Items Listed as Unavailable (is_available = 0):
SELECT r.name
FROM Restaurants r
WHERE EXISTS (
   SELECT 1
   FROM Menu_Items mi
   WHERE mi.restaurant_id = r.restaurant_id
     AND mi.is_available = 0
);
-- Query 114.Find Customers Who Have Ordered from the Restaurant "Tasty Bites":
SELECT c.name
FROM Customers c
WHERE EXISTS (
   SELECT 1
   FROM Orders o
   JOIN Restaurants r ON o.restaurant_id = r.restaurant_id
   WHERE o.customer_id = c.customer_id
     AND r.name = 'Tasty Bites'
);
-- Query 115. count of cuisines total
select count(cuisine_id) as total
from cuisines
where name='italian'
-- Query 116. Insert multiple users
INSERT INTO Users (email, password, role) VALUES ('user1@example.com', 'password1', 'customer'),
```

('user2@example.com', 'password2', 'restaurant');

select*from users

-- **Query** 117. Insert multiple customers

select distinct customer_id from customers

-- **Query** 118. max adress from restaurants

select max(address) as total

from restaurants

-- **Query** 119. Insert multiple menu items

INSERT INTO Menu_Items (restaurant_id, name, description, price, image_url) VALUES (1, 'Sushi Roll', 'Fresh sushi roll', 10.99, 'http://example.com/sushi.jpg'), (2, 'Curry', 'Spicy curry dish', 8.99, 'http://example.com/curry.jpg');

select*from menu_items

-- **Query** 120. Clean up unused users

DELETE FROM Users WHERE user_id NOT IN (SELECT DISTINCT user_id FROM Customers);

-- **Query** 121. Clean up unused cuisines

DELETE FROM Cuisines WHERE cuisine_id NOT IN (SELECT DISTINCT cuisine_id FROM Restaurants);

-- **Query** 122. Clean up unused menu items

DELETE FROM Menu_Items WHERE item_id NOT IN (SELECT DISTINCT item_id FROM Order_Items);

-- **Query** 123. Clean up unused complaints

DELETE FROM Complaints WHERE complaint_id IN (SELECT DISTINCT complaint_id FROM Orders);

-- **Query** 124. Clean up unused coupons

DELETE FROM Coupons WHERE coupon_id in(SELECT DISTINCT coupon_id FROM Payments);

-- **Query** 125. Retrieve the top 5 restaurants by total sales

SELECT r.name, SUM(o.total_amount) AS total_sales FROM Restaurants r JOIN Orders o ON r.restaurant_id = o.restaurant_id GROUP BY r.name ORDER BY total_sales DESC;

-- **Query** 126. Retrieve the top 5 customers by total spending

SELECT c.name, SUM(o.total_amount) AS total_spent FROM Customers c JOIN Orders o ON c.customer_id = o.customer_id GROUP BY c.name ORDER BY total_spent DESC;

-- **Query** 127. Retrieve the top 5 delivery agents by total deliveries

SELECT da.name, COUNT(d.delivery_id) AS total_deliveries FROM Delivery_Agents da JOIN Deliveries d ON da.agent_id = d.agent_id GROUP BY da.name ORDER BY total_deliveries DESC ;

-- **Query** 128. Retrieve the top 5 most reviewed restaurants

SELECT r.name, COUNT(re.review_id) AS total_reviews FROM Restaurants r JOIN Reviews re ON r.restaurant_id = re.restaurant_id GROUP BY r.name ORDER BY total_reviews DESC ;

-- **Query** 129. Retrieve the top 5 most popular menu items

SELECT m.name, SUM(oi.quantity) AS total_ordered FROM Menu_Items m JOIN Order_Items oi ON

m.item_id = oi.item_id GROUP BY m.name ORDER BY total_ordered DESC ;

-- **Query** 130. Check for orphaned orders without customers

SELECT * FROM Orders WHERE customer_id NOT IN (SELECT customer_id FROM Customers);

-- **Query** 131. Check for orphaned menu items without restaurants

SELECT * FROM Menu_Items WHERE restaurant_id NOT IN (SELECT restaurant_id FROM Restaurants);

-- **Query** 132. Check for orphaned reviews without customers

SELECT * FROM Reviews WHERE customer_id IN (SELECT customer_id FROM Customers);

-- **Query** 133. Check for orphaned complaints without orders

SELECT * FROM Complaints WHERE order_id NOT IN (SELECT order_id FROM Orders);

-- **Query** 134. Check for orphaned deliveries without orders

SELECT * FROM Deliveries WHERE order_id NOT IN (SELECT order_id FROM Orders);

-- **Query** 135. Retrieve all orders for a specific customer

SELECT * FROM Orders WHERE customer_id = 1;

-- **Query** 136. Retrieve all reviews left by a specific customer

SELECT * FROM Reviews WHERE customer_id = 1;

-- **Query** 137. Retrieve all wishlist items for a specific customer

SELECT * FROM Wishlists WHERE customer_id = 1;

-- **Query** 138. Retrieve all complaints made by a specific customer

SELECT * FROM Complaints WHERE customer_id = 1;

-- **Query** 139. Retrieve all deliveries assigned to a specific delivery agent

SELECT * FROM Deliveries WHERE agent_id = 1;

-- **Query** 140. Generate a report of total sales by month

SELECT MONTH(order_date) AS month, SUM(total_amount) AS total_sales FROM Orders GROUP BY MONTH(order_date);

-- **Query** 141. Generate a report of total orders by month

SELECT MONTH(order_date) AS month, COUNT(order_id) AS total_orders FROM Orders GROUP BY MONTH(order_date);

-- **Query** 142. Generate a report of total reviews by month

SELECT MONTH(review_date) AS month, COUNT(review_id) AS total_reviews FROM Reviews GROUP BY MONTH(review_date);

-- **Query** 143. Generate a report of total complaints by month

SELECT MONTH(complaint_date) AS month, COUNT(complaint_id) AS total_complaints FROM Complaints GROUP BY MONTH(complaint_date);

-- **Query** 144. Generate a report of total active users by month

SELECT MONTH(created_at) AS month, COUNT(user_id) AS total_users FROM Users GROUP BY MONTH(created_at);

-- **Query** 145. total sum of price

select sum(price) as total

```
from order_items
-- Query 146. count of payments and amount
select count(payment_id) as payment_total,count(amount) as total
from payments
-- Query 147. average price from menuitems
select avg(price) as total
from menu_items
group by item_id
-- Query 148. distinct user_id from customers
select distinct user_id from customers
-- Query 149. total names of resturants
select count(name) as total from Restaurants
-- Query 150.  old restaurant working hours
DELETE FROM Restaurant_Working_Hours WHERE is_closed = 1;
select*from Restaurant_Working_Hours;
```

CHAPTER 4. CONCLUSION AND FUTUREWORK

4.1 Conclusion

A well-designed online food delivery system offers numerous advantages, similar to an online booking system. By leveraging a database-driven platform, food delivery businesses can enhance efficiency, reduce manual work, and optimize customer experience. The key outcomes and benefits of such a system include:

Increased Sales – A food delivery system enables customers to place orders 24/7, boosting revenue and attracting a wider audience. Offering multilingual support and local currency payment options can further enhance market reach.

Reduced Telephone-Based Orders – Automating the ordering process minimizes reliance on call-based orders, reducing the need for extensive customer support and lowering operational costs.

Streamlined and Standardized Payments – Secure online payment processing ensures that transactions are instant and hassle-free, eliminating cash-handling risks and improving financial management.

Improved Process Management – The system automates order processing, manages restaurant inventory, tracks deliveries, and reduces manual errors, leading to greater efficiency.

Streamlined Reporting – Integrated dashboard and analytics tools provide insights into sales trends, customer preferences, and overall business performance, helping businesses refine their strategies.

4.2 Future Work

The future of **food delivery systems** will be shaped by advancements in database management, automation, and artificial intelligence. **Implementing AI-powered personalization** can enhance user experience by recommending food based on customer preferences and past orders. Cloud-based databases will improve data accessibility and security, ensuring seamless real-time order management. Secure transactions can be ensured through blockchain-based payment systems, reducing fraud risks. Additionally, real-time inventory tracking can optimize stock levels, minimizing food wastage in restaurants. The integration of IoT devices can streamline order processing and enable efficient tracking of deliveries. To further improve delivery speed and efficiency, autonomous delivery options such as drones and robots may be explored. The adoption of voice assistants and AI chatbots will allow customers to place orders effortlessly. Big data analytics will help businesses predict demand, optimize delivery routes, and refine restaurant operations. Sustainability efforts, such as eco-friendly packaging and electric vehicle

(EV) delivery fleets, will promote environmental responsibility. Furthermore, Augmented Reality (AR) menu previews can enhance customer decision-making by allowing them to visualize food before ordering. By integrating these innovations, food delivery systems will become more efficient, intelligent, and customer-centric, ensuring a seamless and sustainable future.